15º Congresso de Inovação, Ciência e Tecnologia do IFSP - 2024

AUTOBEO: UMA SOLUÇÃO PARA CONSTRUÇÃO DE UM CLUSTER BEOWULF

EDER L. S. FILHO[1], VINICIUS H. NASCIMENTO[2], RENATO C. BARROS[3]

[1] Graduando em Engenharia da Computação, IFSP, Câmpus Birigui, eder.filho@aluno.ifsp.edu.br.
[2] Graduando em Engenharia da Computação, IFSP, Câmpus Birigui, vinicius.henrique@aluno.ifsp.edu.br.
[3] Doutor em Agronomia com ênfase em Sistemas da Informação, IFSP, Câmpus Birigui, rbarros@ifsp.edu.br

Área de conhecimento (Tabela CNPq): 1.03.01.01-1 Computabilidade e Modelos de Computação.

RESUMO: Este projeto tem como objetivo automatizar o processo de configuração de um Cluster Beowulf. A metodologia adotada consistiu em mapear e estudar o processo de configuração manual, máquina a máquina, para identificar pontos de otimização e automatização. Utilizando um sistema operacional Ubuntu modificado, com as dependências necessárias, e um script em Python estruturado em etapas, foi possível automatizar e reduzir consideravelmente o tempo de configuração de maneira amigável para o usuário, substituindo grande parte da intervenção humana.

PALAVRAS-CHAVE: Cluster Beowulf; automação; otimização; script; Ubuntu; Python.

AUTOBEO: A SOLUTION FOR BUILDING A BEOWULF CLUSTER

ABSTRACT: This project aims to automate the configuration process of a Beowulf cluster. The methodology adopted involved mapping and studying the manual configuration process, machine by machine, to identify optimization and automation points. By using a modified Ubuntu operating system with the necessary dependencies and a Python script structured in stages, it was possible to automate and significantly reduce the configuration time in a user-friendly manner, replacing much of the manual intervention.

KEYWORDS: Cluster Beowulf; automation; optimization; script; Ubuntu; Python.

INTRODUÇÃO

Nas últimas décadas, o poder de processamento dos microprocessadores cresceu cerca de 40% ao ano, em contraste com o crescimento de 20% ao ano observado em minicomputadores e supercomputadores. Esse avanço acelerado deve-se à evolução contínua em arquitetura e tecnologia dos microprocessadores, incluindo melhorias significativas como a introdução da memória cache, que reduz o tráfego no barramento de memória. Com o tempo, o ritmo das inovações tecnológicas desacelerou e as melhorias tornaram-se mais graduais. Para superar essas limitações, a comunidade científica começou a explorar novas abordagens, sendo o paralelismo uma das mais promissoras. Anteriormente visto como uma área

exótica da computação, o paralelismo tornou-se essencial na programação de ambientes como clusters (PITANGA, 2002).

Clusters são grupos de computadores interconectados que operam em conjunto como um único sistema, visando aumentar a capacidade de processamento e a confiabilidade (PACHECO, 2011). Atualmente muitas instituições científicas utilizam clusters para processar grandes volumes de dados e acelerar a obtenção de resultados em pesquisas e disciplinas científicas. (PITANGA, 2002).

O Cluster Beowulf, desenvolvido pela NASA, tornou o paralelismo mais acessível, utilizando computadores pessoais conectados para aumentar o poder de processamento (PACHECO, 2011). Essa técnica possibilitou a criação de clusters utilizando hardware antigo para obter alto desempenho com baixo custo.

Assim, o objetivo deste trabalho é desenvolver uma solução para automatizar a configuração de um Cluster Beowulf, simplificando sua implementação e tornando-a mais rápida e acessível.

MATERIAIS E MÉTODOS

Os materiais utilizados nesta pesquisa incluem:

Hardware:

- 1 Computador contendo um processador Intel Core I5 12450H (8 núcleos | 12 threads);
- 3 Máquinas virtuais sendo:

  1 mestre (utilizando 2 núcleos de processamento);

  2 nós (cada uma utilizando 2 núcleos de processamento).

Software:

- Sistema Operacional Ubuntu 20.04;
- Oracle VM VirtualBox 6.2.16;
- Python v3.10;
- Penguins' Eggs 10.0.35;
- Servidor DHCP - ISC DHCP 4.4.3-P1;
- Servidor OpenSSH 9.8;
- Servidor de Arquivos - Network File System (NFS) 4.2;
- Open MPI 5.0.5.

Para dar início à execução do projeto, foi elaborado um fluxograma detalhando todo o processo de configuração de um Cluster Beowulf manualmente, que foi cuidadosamente analisado. A partir dessa análise minuciosa, foram definidas as ações descritas nesta seção.

Utilizando uma máquina com o sistema operacional Ubuntu e o software de clonagem de máquinas Penguins' Eggs, foi gerada uma imagem modificada do sistema operacional com todas as dependências necessárias para a configuração.

Para que tudo funcionasse integralmente respeitou-se os seguintes parâmetros:

- Primeiramente, todas as máquinas deviam estar com o sistema operacional modificado devidamente instalado e o nome do usuário deveria ser o mesmo em todas elas, neste caso, utilizou-se "cluster";
- Os computadores deviam estar conectados através de uma rede LAN (Rede de área local). Neste caso, simulou-se uma através do software VirtualBox. A arquitetura de rede utilizada pode ser vista na Figura 1.

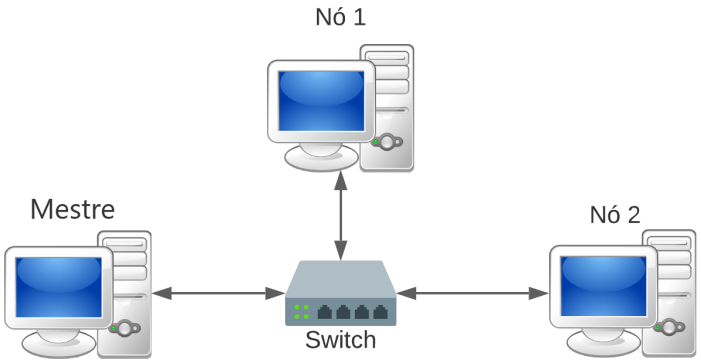


Figura 1: Rede LAN

Pensando na automação da configuração de um Cluster Beowulf baseado na biblioteca Open MPI, foi desenvolvido, utilizando linguagem Python, um script que realiza a configuração completa. Tal script foi executado através da máquina mestre e possui um menu com os as etapas de configuração.

A primeira etapa do script realiza a construção da base da rede, resultado na instalação e configuração do DHCP, os comandos executados linearmente analisam o nome da placa de rede do computador e instalam os pacotes do sistema, limpando os arquivos e escrevendo as configurações básicas para o funcionamento do DHCP. Com este script foi possível montar a rota de distribuição dos IP´s aos demais computadores, tornando automática a distribuição dos mesmos a todos conectados na rede (Figura 2).

```
# SCRIPT PARA CONFIGURAÇÃO DE CLUSTER - DHCP

import os

#Configuração do DHCP

#criando rede
os.system("sudo ifconfig enp0s8 192.168.40.1 netmask 255.255.255.0") #verificar nome da placa

os.system(" > /etc/default/isc-dhcp-server") #limpa o arquivo

arquivo = open("/etc/default/isc-dhcp-server", "a")
dhcp = list()
dhcp.append('INTERFACESv4="enp0s8') #verificar nome da placa
arquivo.writelines(dhcp)

os.system(" > /etc/dhcp/dhcpd.conf") #limpa o arquivo

arquivo = open("/etc/dhcp/dhcpd.conf", "a")
dhcp2 = list()
dhcp2.append('option domain-name "laboratorio.rede";\n')
dhcp2.append('option domain-name-servers teste1.laboratorio.rede;\n')
dhcp2.append('default-lease-time 60;\n')
dhcp2.append('max-lease-time 86400;\n')
dhcp2.append('authoritative;\n\n')

dhcp2.append('subnet 192.168.40.0 netmask 255.255.255.0 {\n')
dhcp2.append('option routers 192.168.40.1;\n')
dhcp2.append('option subnet-mask 255.255.255.0;\n')
dhcp2.append('option domain-search "laboratorio.rede";\n')
dhcp2.append('option domain-name-servers 192.168.40.3, 8.8.8.8;\n')
dhcp2.append('range 192.168.40.20 192.168.40.100;\n')
dhcp2.append('}')
arquivo.writelines(dhcp2)
```

Figura 2: Script DHCP

A segunda etapa é constituída de analises e preparações do "solo", atuando na coleta da lista de

IP´s atribuídos as máquinas, e então inicia-se a configuração do SSH. O script atua iniciando o sistema, e após a verificação do status, gerou-se a chave criptografada de acesso entre os computadores, ao qual a automação irá distribuir e alocar em cada nó verificado anteriormente, assim estabelecendo as rotas de troca de informação (Figura 3).

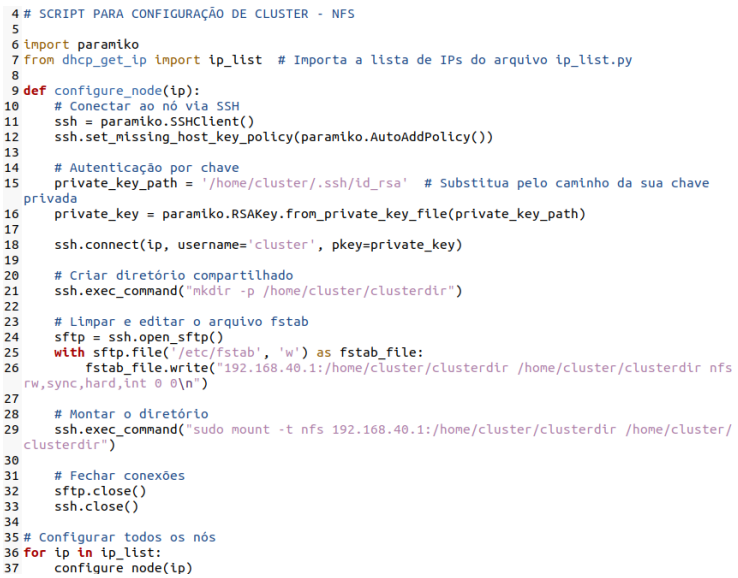
```
# SCRIPT PARA CONFIGURAÇÃO DE CLUSTER - SSH

import os
from dhcp_get_ip import ip_list  # Importa a lista de IPs do arquivo ip_list.py

#start no SSH
print("Iniciando servidor ssh...")
os.system("systemctl enable ssh")

#monitorar o SSH
print("Monitorando servidor ssh...")
os.system("systemctl status ssh")

# Defina o caminho da pasta onde você deseja gerar a chave
ssh_key_dir = "/home/cluster/.ssh"
ssh_key_path = os.path.join(ssh_key_dir, "id_rsa")

# Gerar chave SSH
print(f"Gerando chave ssh em {ssh_key_path}...")
os.system(f"sudo -u cluster ssh-keygen -f {ssh_key_path} -N ''")

# Copia a chave para cada IP na lista
print("Copiando chave para os IPs...")
for ip in ip_list:
    os.system(f"sudo -u cluster ssh-copy-id -i {ssh_key_path}.pub cluster@{ip}")
```

Figura 3: Script SSH

A terceira etapa existe para criar o diretório de arquivos síncrono que são utilizados durante a execução do sistema. O script atua na criação e no estabelecimento das permissões para o acesso via rede, utilizando o sistema de NFS (Network file system), configurando a transição de pacotes entre nós e mestre (Figura 4 e Figura 5).

```
# SCRIPT PARA CONFIGURAÇÃO DE CLUSTER - NFS

import os

#criar diretório compartilhado mestre
os.system("mkdir /home/cluster/clusterdir")

os.system(" > /etc/exports") #limpa o arquivo

#editar exports (apenas Mestre)
arquivo = open("/etc/exports", "a")
exports = list()
exports.append("/home/cluster/clusterdir
192.168.40.0/24(rw,no_subtree_check,async,no_root_squash) \n")
arquivo.writelines(exports)

#iniciar serviço de NFS (apenas Mestre)
print("Iniciando servidor NFS...")
os.system("systemctl enable nfs-kernel-server")

#iniciar serviço de NFS (apenas Mestre)
print("Restart servidor NFS...")
os.system("sudo /etc/init.d/nfs-kernel-server restart")

```

Figura 4: Script NFS - Mestre



```python
# SCRIPT PARA CONFIGURAÇÃO DE CLUSTER - NFS

import paramiko
from dhcp_get_ip import ip_list  # Importa a lista de IPs do arquivo ip_list.py

def configure_node(ip):
    # Conectar ao nó via SSH
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    # Autenticação por chave
    private_key_path = '/home/cluster/.ssh/id_rsa'  # Substitua pelo caminho da sua chave privada
    private_key = paramiko.RSAKey.from_private_key_file(private_key_path)

    ssh.connect(ip, username='cluster', pkey=private_key)

    # Criar diretório compartilhado
    ssh.exec_command("mkdir -p /home/cluster/clusterdir")

    # Limpar e editar o arquivo fstab
    sftp = ssh.open_sftp()
    with sftp.file("/etc/fstab", 'w') as fstab_file:
        fstab_file.write("192.168.40.1:/home/cluster/clusterdir /home/cluster/clusterdir nfs rw,sync,hard,int 0 0\n")

    # Montar o diretório
    ssh.exec_command("sudo mount -t nfs 192.168.40.1:/home/cluster/clusterdir /home/cluster/clusterdir")

    # Fechar conexões
    sftp.close()
    ssh.close()

# Configurar todos os nós
for ip in ip_list:
    configure_node(ip)
```

Figura 5: Script NFS - Nó

Para avaliar o desempenho desta pesquisa em comparação ao processo manual de instalação e configuração de um Cluster Beowulf, foram realizados testes cronometrando o tempo necessário para completar cada etapa.

No processo manual, utilizou-se três máquinas virtuais, simulando o uso típico de um usuário padrão de Linux. Seguindo um guia de criação de um Cluster Beowulf criado previamente, foram realizadas as etapas de instalação do sistema operacional, ajustes nos arquivos de configuração e instalação manual das dependências, utilizando comandos Linux diretamente no terminal. Para configurar cada máquina, foi necessário acessá-las presencialmente, uma a uma, tornando o processo mais demorado e trabalhoso.

Já no processo automatizado, também foram utilizadas três máquinas virtuais. Foi instalado o sistema operacional gerado previamente, contendo todas as dependências necessárias pré-instaladas e o script desenvolvido. O script utilizou o protocolo SSH para acessar e configurar cada máquina remotamente, eliminando a necessidade de ajustes manuais individuais e automatizando todo o procedimento de maneira eficiente.

RESULTADOS E DISCUSSÃO

Com a construção do script foi possível obter uma execução linear do processo de automação, tendo alcançado uma configuração limpa e funcional.

Além disso, os tempos cronometrados durante os processos de instalação e configuração do Cluster Beowulf revelaram diferenças significativas entre os métodos manual e automatizado. Os resultados são apresentados na Tabela 1.

Tabela 1: Tempos de execução

Processo	Sistema Operacional (Horas)	Configuração (Horas)	Total (Horas)
Manual	00:26:24	00:36:36	01:03:00
Automatizado	00:14:18	00:01:01	00:15:19

A automação resultou em uma redução de aproximadamente 76% no tempo total de instalação e configuração. O método automatizado não apenas acelerou o processo, mas também minimizou a

possibilidade de erros humanos e eliminou a necessidade de acesso físico a cada máquina, graças ao uso do protocolo SSH.

Além disso, o processo automatizado diminui o nível de abstração para os usuários que desejam criar um Cluster Beowulf. Com um script simplificado, mesmo aqueles com conhecimentos limitados em administração de sistemas podem realizar a instalação e configuração de forma mais intuitiva, tornando a tarefa mais acessível e menos intimidadora.

CONCLUSÕES

Os resultados indicam que a automação, através de scripts eficientes, pode transformar radicalmente a forma como as configurações de clusters são realizadas, promovendo não apenas rapidez, mas também uma abordagem mais escalável e sustentável em ambientes de computação distribuída. A implementação de soluções automatizadas, como demonstrado neste estudo, não só economiza tempo, mas também melhora a confiabilidade e a repetibilidade dos processos de instalação e configuração.

CONTRIBUIÇÕES DOS AUTORES

Todos os autores contribuíram com a concepção, escopo do estudo e revisão do trabalho aprovaram a versão submetida. Eder L. S. Filho e Vinicius H. Nascimento foram responsáveis pela metodologia, execução dos experimentos e pela escrita. Renato C. Barros contribuiu com a orientação técnica.

AGRADECIMENTOS

Agradecemos ao nosso orientador, Renato C. Barros, por sua valiosa orientação técnica e contribuições ao longo deste trabalho. Expresso um agradecimento especial ao Grupo de Pesquisa em Visão Computacional e Inteligência Artificial - GPeVICIA, pelo apoio e material fornecido para esta pesquisa. Também gostaríamos de agradecer a todos os fóruns sobre Linux que, por meio de suas discussões e compartilhamento de conhecimento, mantêm vivo o espírito colaborativo deste sistema operacional. Por fim, agradecemos a todos que, direta ou indiretamente, contribuíram para o sucesso desta pesquisa.

REFERÊNCIAS

PACHECO, P. S. *An Introduction to Parallel Computing*. Burlington, MA: Elsevier Inc., 2011.

PITANGA, M. J. *Construindo supercomputadores com linux*. 1. ed. Rio de Janeiro: Brasport Livros e Multimídia Ltda., 2002.